CAPG48LSRB347

ARCHANA.S.AJITH

1.What is difference between Overloading and Overriding?

|  |  |
| --- | --- |
| Overloading | Overriding |
| * Overloading implements compile time polymorphism. | * Overriding implements runtime polymorphism. |
| * Overloading occurs between same classes. | * Overriding occurs between superclass and subclass. |
| * Overloading methods have same name but different parameters. | * Overriding methods have same signature i.e same name and method arguments. |
| * With overloading method call is determined at compile time. | * With overriding method call is determined at run time based on object types. |
| * Causes compile time error and easy to fix. | * Causes runtime error if breaks and causes serious problems |

2.If both the functions are static, can we override?

Ans) No, we cannot override static methods because method overriding is based on dynamic binding at runtime and the static methods are bonded using static binding at compile time. So, we cannot override static methods.

The calling of method depends upon the type of object that calls the static method. It means:

* If we call a static method by using the parent class object, the original static method will be called from the parent class.
* If we call a static method by using the child class object, the static method of the child class will be called.

1. Difference between Final, Finally and Finalize?

|  |  |  |
| --- | --- | --- |
| Final | Finally | Finalize |
| * Final is the keyword and access modifier which is used to apply restrictions on a class, method or variable. | * Finally is the block in Java Exception handling to execute the important code whether the exception occurs or not. | * finalize is the method in Java which is used to perform clean up processing just before object is garbage collected. |
| * Final keyword is used with the classes, methods and variables. | * Finally block is always related to the try and catch block in exception handling. | * finalize() method is used with the objects |
| * Once declared, final variable becomes constant and cannot be modified. Final cannot be overridden by subclasses. | * finally block runs the important code even if exception occurs or not. | * finalize method performs the cleaning activities with respect to the object before its destruction. |
| * Final method is executed only when we call it. | * Finally block is executed as soon as the try-catch block is executed. | * finalize method is executed just before the object is destroyed. |

4.What are marker interfaces?

Ans) A marker interface is an interface that has no methods or constants inside it. It provides run-time type information about objects, so the compiler and JVM have additional information about the object.

A marker interface is also called a tagging interface.

Ex. public interface Deletable {

}

5.What is a singleton class?

Ans) In object-oriented programming, a singleton class is a class that can have only one object (instance of class) at a time. After first time, if we try to instantiate the Singleton class, the new variable also points to the first instance created.

Key points to remember while defining Singleton class:

* Make constructor private.
* Write static method that has return type object of this singleton class.

6.What are the underlying data structure for arraylist and linked list?

Ans) ArrayList is a resizable array implementation in java. The backing data structure of ArrayList is an array of Object class.

Linked List underlying data structure is linear data structure.

7.What is hash collision?

Ans) A collision, or more specifically, a hash code collision in a HashMap, is a situation where two or more key objects produce the same final hash value and hence point to the same bucket location or array index.

8.What is lambda expressions?

Ans) A lambda expression is a short block of code which takes in parameters and returns a value. Lambda expressions are similar to methods, but they do not need a name and they can be implemented right in the body of a method.

9.What is functional interface?

Ans) A functional interface is an interface that contains only one abstract method. They can have only one functionality to exhibit. From Java 8 onwards, lambda expressions can be used to represent the instance of a functional interface. A functional interface can have any number of default methods.

10.What are optional classes in java 8?

Ans) Optional is a container object used to contain not-null objects. Optional object is used to represent null with absent value. This class has various utility methods to facilitate code to handle values as 'available' or 'not available' instead of checking null values.

11. What is factory design pattern and Adapter and difference between them?

Ans) The Factory Design Pattern or Factory Method Design Pattern is one of the most used design patterns in Java. According to GoF, this pattern “defines an interface for creating an object, but let subclasses decide which class to instantiate. The Factory method lets a class defer instantiation to subclasses”.

Adapter is a structural design pattern, which allows incompatible objects to collaborate. The Adapter acts as a wrapper between two objects. It catches calls for one object and transforms them to format and interface recognizable by the second object.

12. What are solid principles?

Ans) SOLID principles are object-oriented design concepts relevant to software development. SOLID is an acronym for five other class-design principles: Single Responsibility Principle, Open-Closed Principle, Liskov Substitution Principle, Interface Segregation Principle, and Dependency Inversion Principle.

13.What does @Component and @Scan do in Springboot annotation?

Ans) @Component = It is a class-level annotation that turns the class into Spring bean at the auto-scan time.

@Scan = It is used to specify packages to locate components in Spring. This is automatically done in the default package, in the case of Spring Boot.

14. What is the difference between JPA, Hibernate and ORM?

Ans) ORM: Object Relational Mapping is concept/process of converting the data from Object oriented language to relational DB and vice versa. For example in java its done with the help of reflection and jdbc.

Hibernate: Its the implementation of above concept.

JPA: Its the one step above ORM. Its high level API and specification so that different ORM tools can implement so that it provides the flexibility to developer to change the implementation from one ORM to another (for example if application uses the JPA api and implementation is hibernate. In future it can switch to IBatis if required. But on the other if application directly lock the implementation with Hibernate without JPA platform, switching is going to be herculean task)

15. What is indexing in database?

Ans) Indexing is a way to optimize the performance of a database by minimizing the number of disk accesses required when a query is processed. It is a data structure technique which is used to quickly locate and access the data in a database. Indexes are created using a few database columns.

16.What are four types of drivers in JDBC?

Ans: Four types of drivers in JDBC are:

Type 1: JDBC-ODBC bridge

Type 2: partial Java driver

Type 3: pure Java driver for database middleware

Type 4: pure Java driver for direct-to-database

17.What is difference between @component and @bean and @service?

Ans) i).It is used to explicitly declare a single bean, rather than letting Spring do it automatically.

ii).If any class is annotated with @Component it will be automatically detect by using classpath scan.

iii).We should use @bean, if you want specific implementation based on dynamic condition.

iv).The @Service is child of component and used to denote classes from the service layer of the application.

v).@Component is a generic stereotype for any Spring-managed component. @Service annotates classes at the service layer.

vi).@bean annotates classes at the persistence layer, which will act as a database repository.

18.What is difference between SOAP and restful services technically?

Ans) i).SOAP stands for Simple Object Access Protocol whereas REST stands for Representational State Transfer.

ii).SOAP is a protocol whereas REST is an architectural pattern.

iii).SOAP uses service interfaces to expose its functionality to client applications while REST uses Uniform Service locators

to access to the components on the hardware device.

iv).SOAP needs more bandwidth for its usage whereas REST doesn’t need much bandwidth.

v).Comparing SOAP vs REST API, SOAP only works with XML formats whereas REST work with plain text, XML, HTML and JSON.

vi).SOAP cannot make use of REST whereas REST can make use of SOAP.

19.Difference in POST And PUT.

Ans)i).PUT - This method is idempotent.

POST - This method is not idempotent.

ii)PUT method is call when you have to modify a single resource, which is already a part of resource collection.

POST method is call when you have to add a child resource under resources collection.

iii).PUT method answer can be cached.

You cannot cache PUT method responses.

iv).PUT /vi/juice/orders/1234 indicates that you are updating a resource which is identified by “1234”.

POST /vi/juice/orders indicates that you are creating a new resource and return an identifier to describe the resource.

v).PUT works as specific.

POST work as abstract.

vi).We use UPDATE query in PUT.

We use create query in POST.

vii).In PUT method, the client decides which URI resource should have.

In POST method, the server decides which URI resource should have.

20.Is React uni directional or bi-directional?

Ans) React doesn't support bi-directional binding to make sure you are following a clean data flow architecture.

The major benefit of this approach is that data flows throughout your app in a single direction, giving you better control over it.

In terms of React it means: state is passed to the view and to child components.

React, a Javascript library, uses unidirectional data flow.

The data from the parent is known as props.

You can only transfer data from parent to child and not vice versa.

This means that the child components cannot update or modify the data on their own, making sure that a clean data flow architecture is followed.